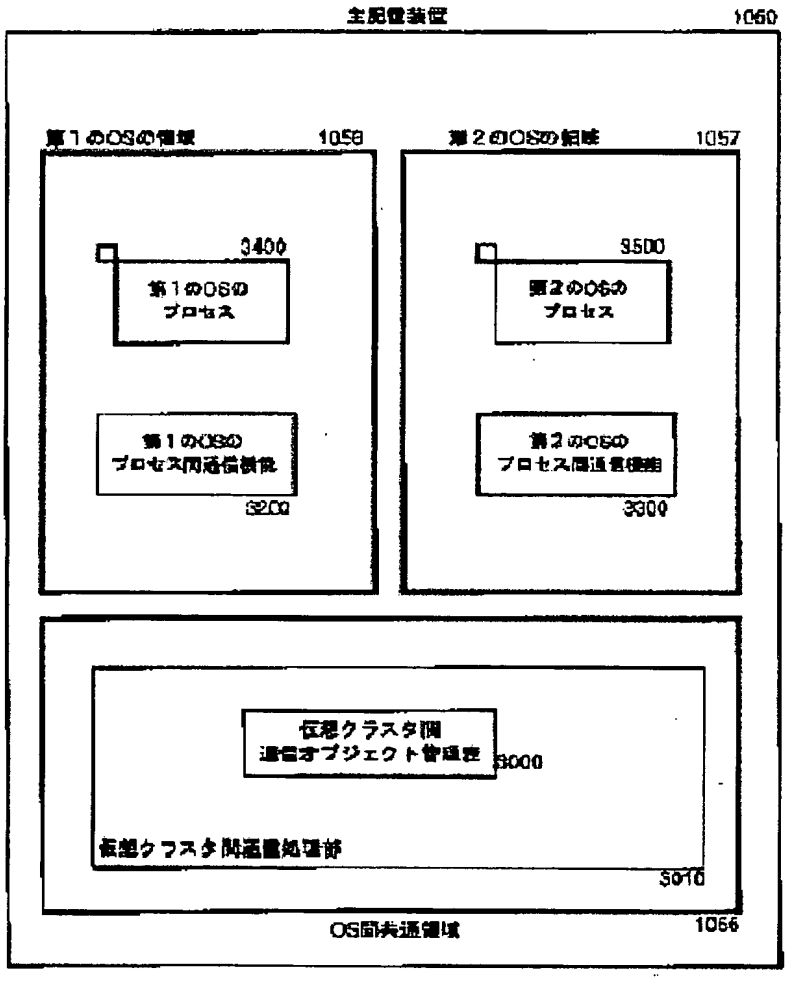
Home Search List ☐ Include

MicroPatent® PatSearch FullText: Record 1 of 1

Search scope: JP (bibliographic data only)
Years: 1991-2003
Patent/Publication No.: JP11085547



Order This Patent | Family Lookup | Find Similar | Legal Status

Go to first matching text

JP11085547 A
VIRTUAL CLUSTER CONFIGURATION METHOD
HITACHI LTD
Inventor(s): ;INOUE TARO ;ARAI TOSHIAKI
Application No. 09248179 JP09248179 JP, Filed 19970912,A1 Published 19990330

Abstract: PROBLEM TO BE SOLVED: To reduce exclusive control overhead in a high multiple symmetrical multiprocessor system(SMP) by providing a process that operates on a different virtual cluster with an inter- virtual cluster communicating means which mutually communicates.

SOLUTION: Two virtual clusters are configured on an SMP that has two processors, and a main storage device 1050 consists of an inter-OS common area 1055 and areas 1056 and 1057 for 1st and 2nd OSs. An inter- virtual cluster processing part 3010 exists in the area 1050, and an inter-virtual cluster communication object management table

3000 exists in it. Inter-process communication functions 3200 and 3300 for the 1st and 2nd OSs exist in the areas 1056 and 1057 of the 1st and 2nd OSs respectively. They perform inter-process communication processing to a process on which respective OSs operate, themselves. Respective processes 3400 and 3500 for the 1st and 2nd OSs exist in the areas 1056 and 1057 of the 1st and 2nd OSs.

Int'l Class: G06F00946; G06F01516

Patents Citing this One: No US, EP, or WO patents/search reports have cited this patent.

---

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平11－85547

(43)公開日　平成11年(1999)３月30日

(51)Int.Cl.[6]		識別記号	FI		
Ｇ０６Ｆ	9/46	３５０	Ｇ０６Ｆ	9/46	３５０
	15/16	４３０		15/16	４３０Ｃ

審査請求　未請求　請求項の数15　ＯＬ　（全 22 頁）

(21)出願番号　特願平9－248179

(22)出願日　平成９年(1997)９月12日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　井上　太郎
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(72)発明者　新井　利明
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(74)代理人　弁理士　小川　勝男

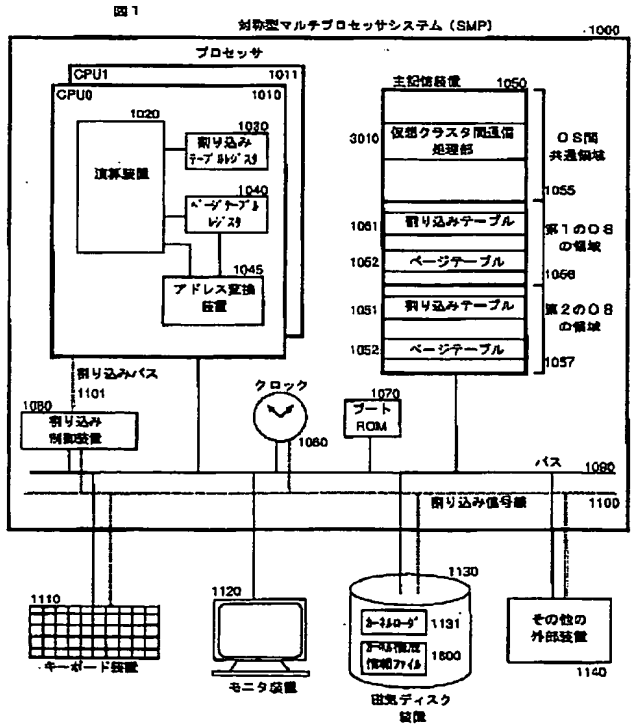
(54)【発明の名称】　仮想クラスタ構成方法

(57)【要約】

【課題】高多重の対称型マルチプロセッサシステムにおいて、排他制御オーバヘッドおよびキャッシュミス率を小さくする。

【解決手段】１台の計算機の上に、プロセッサ、主記憶およびデバイスを専有しその上でＯＳが動作する仮想クラスタを複数個構築し、仮想クラスタ上のプロセス間の通信機能により仮想クラスタ間の処理の連携を可能とする。

図１



【特許請求の範囲】

【請求項１】プロセッサ、物理メモリ、外部デバイス等のハードウェア資源を専有しその上でオペレーティングシステムが動作する複数個の仮想クラスタを１台の計算機上に構成する仮想クラスタ構成手段を有し、
異なる仮想クラスタ上で動作するプロセスが相互に通信する仮想クラスタ間通信手段を有することを特徴とする仮想クラスタ構成方法。

【請求項２】請求項１の仮想クラスタ構成方法において、
当該仮想クラスタ構成手段は、第１の仮想クラスタのオペレーティングシステム（第１のオペレーティングシステム）の起動時に、他の仮想クラスタが利用するハードウェア資源（プロセッサ、物理メモリ、外部デバイス）を他の仮想クラスタのために予約して第１のオペレーティングシステムの管理対象から外し第１のオペレーティングシステムからはアクセス不可能にする手順を有し、予約された物理メモリ領域に第２の仮想クラスタのオペレーティングシステム（第２のオペレーティングシステム）をロードして起動する手順を有することを特徴とする仮想クラスタ構成方法。

【請求項３】請求項２の仮想クラスタ構成方法において、
仮想クラスタが専有するデバイスからの割り込みを、当該仮想クラスタが専有するプロセッサに通知する手段を有することを特徴とする仮想クラスタ構成方法。

【請求項４】請求項１の仮想クラスタ構成方法において、
前記の仮想クラスタ間通信手段は、仮想クラスタ間通信オブジェクトの名前と、第１のオペレーティングシステムのプロセス間通信機能におけるプロセス間通信オブジェクトハンドルと、第２のオペレーティングシステムのプロセス間通信機能におけるプロセス間通信オブジェクトハンドルとの対応を管理する対応管理手段を有することを特徴とする仮想クラスタ構成方法。

【請求項５】請求項４の仮想クラスタ構成方法において、
第１のオペレーティングシステム上のプロセスが、第２のオペレーティングシステム上のプロセスとの通信要求を発行した場合、当該通信要求を前記対応管理手段を用いて第１のオペレーティングシステムのプロセス間通信機能への通信要求、あるいは第２のオペレーティングシステムのプロセス間通信機能への通信要求に変換する手順を有することを特徴とする仮想クラスタ構成方法。

【請求項６】請求項１の仮想クラスタ構成方法において、
ある仮想クラスタ上で動作するプロセスを、他の仮想クラスタへ移動させて動作させるプロセスマイグレーション手段を有することを特徴とする仮想クラスタ構成方法。

【請求項７】請求項６の仮想クラスタ構成方法において、
前記のプロセスマイグレーション手段は、各仮想クラスタのプロセッサ利用率の情報を得る手順と、
得られたプロセッサ利用率の情報から、最もプロセッサ利用率が低い仮想クラスタをプロセスのマイグレーション先に選択する手順とを有することを特徴とする仮想クラスタ構成方法。

【請求項８】請求項６の仮想クラスタ構成方法において、
前記のプロセスマイグレーション手段は、マイグレーション対象プロセスの親プロセスが動作する仮想クラスタをプロセスのマイグレーション先に選択する手順を有することを特徴とする仮想クラスタ構成方法。

【請求項９】請求項６の仮想クラスタ構成方法において、
前記のプロセスマイグレーション手段は、マイグレーション対象プロセスの親プロセスが動作するのとは異なる仮想クラスタをプロセスのマイグレーション先に選択する手順を有することを特徴とする仮想クラスタ構成方法。

【請求項１０】請求項６の仮想クラスタ構成方法において、
前記のプロセスマイグレーション手段は、各仮想クラスタの実メモリ利用率の情報を得る手順と、
得られた実メモリ利用率の情報から、最も実メモリ利用率が低い仮想クラスタをプロセスのマイグレーション先に選択する手順とを有することを特徴とする仮想クラスタ構成方法。

【請求項１１】請求項６の仮想クラスタ構成方法において、
前記のプロセスマイグレーション手段は、マイグレーション先の仮想クラスタをランダムに選択する手順を有することを特徴とする仮想クラスタ構成方法。

【請求項１２】請求項６の仮想クラスタ構成方法において、
前記のプロセスマイグレーション手段は、マイグレーション先の仮想クラスタをラウンドロビンに選択する手順を有することを特徴とする仮想クラスタ構成方法。

【請求項１３】請求項１の仮想クラスタ構成方法において、
仮想クラスタ間でデバイスを共有するデバイス共有手段を有することを特徴とする仮想クラスタ構成方法。

【請求項１４】請求項１の仮想クラスタ構成方法において、
仮想クラスタ間でメモリを共有する手段を有することを特徴とする仮想クラスタ構成方法。

【請求項１５】請求項１の仮想クラスタ構成方法において、
ある１つの仮想クラスタを別の仮想クラスタのホットス

タンバイとして動作させる手段を有することを特徴とする仮想クラスタ構成方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、計算機の制御方法に関わる。特に、１台計算機上で、複数のオペレーティングシステムを動作させ、各々のオペレーティングシステム上のプロセス同士が通信する方法に関する。

【０００２】

【従来の技術】最近の計算機システムでは、処理能力の向上を図るために複数個のプロセッサ（ＣＰＵ）を搭載するマルチプロセッサ構成が一般的となっている。そのマルチプロセッサシステムの構成方式として、対称型マルチプロセッサシステム（ＳＭＰ）とクラスタ型システムがある。

【０００３】ＳＭＰは、システムを構成するすべてのＣＰＵがメモリを共有し、すべての論理的な資源（プロセス；スレッド等）および物理的な資源（ＣＰＵ、物理メモリ、外部デバイス等）を共有する。そして、１つのオペレーティングシステム（ＯＳ）がシステム全体を制御する。

【０００４】一方、クラスタ型システムでは、システム全体が複数個のクラスタに分割される。各クラスタはＣＰＵ、メモリ、デバイスで構成され、各クラスタ上のＯＳが当該クラスタを制御する。この場合は、論理的な資源、物理的な資源の双方とも共用はされず、完全に分離される。

【０００５】また、従来から、１台の計算機上に複数の仮想的な計算機（仮想計算機：ＶＭ）を実現する仮想計算機システムがある。仮想計算機システムでは、仮想計算機制御プログラムが計算機のすべての資源を管理し仮想化することにより仮想計算機を構成する。仮想計算機システムを実現する場合、すべての計算機資源の仮想化を行うため、ハードウェアおよびソフトウェア（仮想計算機制御プログラム）の規模が大きくなるという問題がある。

【０００６】

【発明が解決しようとする課題】先に述べたように、対称型マルチプロセッサシステム（ＳＭＰ）では、すべてのシステムの資源を共用する。例えば、メモリはすべてのＣＰＵから共有されるので、複数のＣＰＵからアクセスされる可能性があるメモリ領域については排他制御が必要となる。このオーバヘッドはＣＰＵ数のベキ乗に比例して増加するため、高多重ＳＭＰの実現の上での大きな問題となる。

【０００７】そこで、ＳＭＰのハードウェア資源を分割専有する仮想クラスタシステムであれば、ＳＭＰにおけるＣＰＵ数が増加しても排他制御オーバヘッドの増加を小さくできる。例えば、８台のＣＰＵを有するＳＭＰをそのまま使用する場合と、４台のＣＰＵを専有する２つの仮想クラスタに分割する場合の排他制御オーバヘッドを比べると、先に述べたように、排他制御オーバヘッドはＣＰＵ数のベキ乗に比例するので、２つの仮想クラスタに分割した場合の方が排他制御オーバヘッドは小さくなる。

【０００８】本発明の第１の目的は、高多重の対称型マルチプロセッサシステム（ＳＭＰ）において、排他制御オーバヘッドを小さくするために、仮想クラスタを構成する方法を提供することにある。

【０００９】ＳＭＰの別の問題点は、バスの利用率である。バスには各種のデータが流れる。キャッシュミスが発生した場合のメモリからキャッシュへのデータ、各ＣＰＵのキャッシュの内容の不一致を起こさないようにするためのコヒーレントトランザクション等である。ＳＭＰの性能が飽和する原因としては、バスの利用率が過大になるバスネックの発生がある。バスネックの発生を防止するためには、先に挙げたようなバスの利用を減らしてバスの利用率を低くする必要がある。そこで、仮想クラスタを構成すれば、プロセスが動作するプロセッサが限定されるので、キャッシュミスの発生を減少できる。また、メモリは分割専有されるために他の仮想クラスタからのメモリアクセスはないので、コヒーレントトランザクションの波及する範囲を当該仮想クラスタ中に限定できる。

【００１０】本発明の第２の目的は、高多重の対称型マルチプロセッサシステムにおいて、キャッシュミス、およびコヒーレントトランザクションを減らすために、仮想クラスタを構成する方法を提供することにある。

【００１１】上記のように仮想クラスタを構成した場合、仮想クラスタ同士が連携して処理をすすめようとすると、仮想クラスタ間でのプロセスの通信が必要となる。

【００１２】本発明の第３の目的は、仮想クラスタシステムにおいて、各仮想クラスタ上のプロセスが相互に通信する方法を提供することにある。

【００１３】

【課題を解決するための手段】本発明では、第１の仮想クラスタのオペレーティングシステム（第１のオペレーティングシステム）の起動時に、他の仮想クラスタが利用するハードウェア資源を他の仮想クラスタのために予約して第１のオペレーティングシステムの管理対象から外して第１のオペレーティングシステムからはアクセス不可能にすることにより、１台の計算機において複数個の仮想クラスタを、特別なハードウェアなしに構成することが可能となる。

【００１４】また、本発明では、仮想クラスタ間通信オブジェクトの名前と、第１のオペレーティングシステムのプロセス間通信機能におけるプロセス間通信オブジェクトハンドルと、第２のオペレーティングシステムのプロセス間通信機能におけるプロセス間通信オブジェクト

ハンドルとの対応管理手段を設ける。この対応管理手段により、第１のオペレーティングシステム上のプロセスおよび第２のオペレーティングシステム上のプロセスから発行された仮想クラスタ間通信オブジェクトの名前を指定した通信要求を、第１のオペレーティングシステムあるいは第２のオペレーティングシステムのプロセス間通信機能への要求へと変換できる。これにより、例えば、イベント同期によるプロセス間通信の場合では、ウェイトしている第１のオペレーティングシステム上のプロセスを第２のオペレーティングシステム上のプロセスがポストし、ウェイトしている第２のオペレーティングシステム上のプロセスを第１のオペレーティングシステム上のプロセスがポストすることが可能となる。

【００１５】

【発明の実施の形態】以下、本発明の実施例を、図面により詳細に説明する。

【００１６】以下に説明する実施例では、２台のプロセッサを持つ対称型マルチプロセッサシステム（ＳＭＰ）上で、２つの仮想クラスタを構成し、各々の仮想クラスタは１個ずつのプロセッサを専有し、その上では第１のＯＳおよび第２のＯＳが稼働するものとする。もちろん、設定する仮想クラスタ数はいくつでも構わない。ただし、対称型マルチプロセッサシステムが持つプロセッサの個数より多くの仮想クラスタを構築することはできない。

【００１７】図１は、本発明の実施例を実施する計算機システムである対称型マルチプロセッサシステム１０００の構成を示す図である。対称型マルチプロセッサシステム１０００は、プロセッサ１０１０、プロセッサ１０１１、主記憶装置１０５０、クロック１０６０、ブートＲＯＭ１０７０、割り込み制御装置１０８０、バス１０９０、割り込み信号線１１００、および割り込みバス１１０１から成り、キーボード装置１１１０、モニタ装置１１２０、磁気ディスク装置１１３０、その他の外部装置１１４０が接続される。

【００１８】割り込み信号線１１００は、割り込み制御装置１０８０と各種の割り込みを発生する装置を接続する。割り込みが発生すると、割り込み信号線１１００を経由して割り込み制御装置１０８０に通知され、割り込み制御装置１０８０はそれを数値化して、割り込みバス１１０１によりプロセッサ１０１０およびプロセッサ１０１１に通知する。

【００１９】割り込み制御装置１０８０は、外部の装置やクロック１０６０からの割り込み要求信号を数値化してプロセッサ１０１０およびプロセッサ１０１１に伝える。プロセッサからの指示により特定の装置からの割り込みをプロセッサに通知しないようにすることもできる。

【００２０】クロック１０６０は、周期的にクロック割り込みを発生する。

【００２１】ブートＲＯＭ１０７０には、この対称型マルチプロセッサシステム１０００が起動された時の起動手順の最初の一部が記録されている。

【００２２】プロセッサ１０１０およびプロセッサ１０１１は、各々、演算装置１０２０、割り込みテーブルレジスタ１０３０、ページテーブルレジスタ１０４０、およびアドレス変換装置１０４５から成る。以下では、プロセッサ１０１０はＣＰＵ０、プロセッサ１０１１はＣＰＵ１として示す場合がある。この図では、プロセッサはプロセッサ１０１０およびプロセッサ１０１１の２台のみが書かれているが、何台でもかまわない。割り込みテーブルレジスタ１０３０は、当該プロセッサを専有する仮想クラスタのＯＳの割り込みテーブル１０５１のアドレスを格納している。ページテーブルレジスタ１０４０は、当該プロセッサを専有する仮想クラスタのＯＳのページテーブル１０５２のアドレスを格納している。アドレス変換装置１０４５は、演算装置１０２０からアドレスを受け取り、ページテーブルレジスタ１０４０が示すページテーブル１０５２に基づいて仮想アドレスから実アドレスへの変換を行う。演算装置１０２０は変換された実アドレスを用いて主記憶装置１０５０へアクセスする。

【００２３】対称型マルチプロセッサシステム１０００には、外部装置として、キーボード装置１１１０、モニタ装置１１２０、磁気ディスク装置１１３０、その他の外部装置１１４０が接続されており、モニタ装置１１２０を除いては、割り込みバス１１００に接続される。

【００２４】主記憶装置１０５０には、第１のオペレーティングシステム（ＯＳ）の領域１０５６、第２のＯＳの領域１０５７、およびＯＳ間共通領域１０５５がある。第１のＯＳの領域１０５６および第２のＯＳの領域１０５７には、それぞれのＯＳのページテーブル１０５２および割り込みテーブル１０５１がある。ＯＳ間共通領域１０５５には、仮想クラスタ間通信処理部３０１０がある。

【００２５】図２は、ページテーブル１０５２の構成を示す図である。ページテーブル１０５２は仮想アドレス空間単位に存在し、仮想ページ番号１２００毎にエントリがあり、そのエントリは有効ビット１２１０および物理ページ番号１２２０で構成される。有効ビット１２１０は当該仮想ページに対して物理ページが割り当てられているか否かを示す。図２の場合、仮想ページ１は有効ビット１２１０がセットされているので、物理ページが割り当てられているが、仮想ページ４は有効ビット１２１０がセットされていないので、物理ページが割り当てられていない。物理ページ番号１２２０は当該仮想ページに割り当てられている物理ページの番号を示す。

【００２６】アドレス変換装置１０４５は、演算装置１０２０から仮想アドレスを受け取り、ページテーブルレジスタ１０４０が示すページテーブル１０５２の内容に

基づいて仮想アドレスから実アドレスへの変換を行い、得られた実アドレスを演算装置１０２０へ返す。演算装置１０２０はこの実アドレスを用いて主記憶装置１０５０へアクセスする。

【００２７】ページテーブルレジスタ１０４０によりページテーブルを切り替えることで、独立したアドレス空間を構築できる。即ち、第１のＯＳの空間および第２のＯＳの空間を構築できる。そして、各々のＯＳのページテーブルにおいて、共通領域に対応する仮想ページに対して同じ物理ページを設定すればＯＳ間共通領域を実現できる。

【００２８】図３は、割り込みテーブル１０５１の構成を示す図である。割り込みテーブル１０５１は、割り込み番号とそれに対する割り込み処理ハンドラのアドレスの対応を格納する。割り込みテーブル１０５１は、割り込み番号１３００毎にエントリがあり、そのエントリは割り込みハンドラアドレス１３１０で構成される。制御装置１０８０から割り込みを受けたプロセッサは、割り込みテーブルレジスタ１０３０の内容が示す割り込みテーブル１０５１の中から割り込み番号１３００に対応する割り込みハンドラアドレス１３１０を得て、そこに制御を渡して割り込み処理を行う。

【００２９】図４は、計算機のブート時に実行されるカーネルローダ１１３１の処理のフローである。カーネルローダ１１３１は、システムの起動時にブートＲＯＭ１０７０に格納される起動プログラムにより磁気ディスク装置１１３０から主記憶にロードされ実行を開始する。まず、主記憶リスト１７１０、デバイスリスト１７２０、ロードオブジェクトリスト１７４０を初期化する（ステップ１４１０）。これらのリストの詳細は後で説明する。カーネルが使用するページテーブル領域を割り当てる（ステップ１４２０）。この計算機に接続される計算機のハードウェア構成を検査し（ステップ１４３０）、それに基づいてデバイスリスト１７２０を作成する（ステップ１４４０）。次に、磁気ディスク装置１１３０に格納されるカーネル構成情報ファイル１６００を読み込んでＯＳのカーネル構成情報１９００とする（ステップ１４５０）。カーネル構成情報１９００の詳細は後で説明する。カーネル構成情報1.９００で指定されたファイルを読み込み、ロードオブジェクトリスト１７４０を追加する（ステップ１４６０）。そして、ページテーブルレジスタ１０３０に構築したページテーブルのアドレスをセットして仮想アドレスモードへ移行し（ステップ１４７０）、構築した主記憶リスト１７１０、デバイスリスト１７２０、ロードオブジェクトリスト１７４０、カーネル構成情報アドレス１７３０から成るパラメータテーブル１７００をパラメータとしてＯＳの初期化処理へ制御を渡す（ステップ１４８０）。

【００３０】図５は、カーネル構成情報ファイル１６００の内容を示す図である。カーネル構成情報ファイル１６００の内容がそのままカーネル構成情報１９００となる。カーネル構成情報ファイル１６００には、オペレーティングシステムが使用するデータを格納している。データにはその種別を示す名前が付けられており、オブジェクトファイル１６１０という名前のエントリにはそのデータ１６３０が格納され、第２のＯＳ１６２０という名前のエントリには第２のＯＳが使用するデータ１６４０が格納される。第２のＯＳが使用するデータ１６４０には、第２のＯＳ（即ち、第２の仮想クラスタ）に割り当てるプロセッサ数、主記憶の大きさ、デバイスの情報等が含まれる。

【００３１】このカーネル構成情報ファイル１６００を書き換えることにより、仮想クラスタの構成を変更することが可能である。

【００３２】図６は、主記憶リスト１７１０、デバイスリスト１７２０、ロードオブジェクトリスト１７４０およびパラメータテーブル１７００の構成を示す図である。パラメータテーブル１７００は、主記憶リスト１７１０、デバイスリスト１７２０、ロードオブジェクトリスト１７４０、およびカーネル構成情報アドレス１７３０で構成される。

【００３３】主記憶リスト１７１０は、主記憶ブロック１７５０のリストである。主記憶ブロック１７５０は、連続した主記憶領域の状況を記述する。ベースアドレス１７６０は当該主記憶ブロックの開始物理アドレスであり、サイズ１７７０はその大きさを示す。利用状況１７８０は当該主記憶ブロックが未使用か否かを示す値が格納され、次エントリ１７８５には次の主記憶ブロックへのポインタが格納される。

【００３４】デバイスリスト１７２０は、デバイスブロック１８００のリストである。デバイスブロック１８００は、カーネルローダが検出したハードウェアの情報が格納される。デバイスタイプ１８１０には当該デバイスの種類が格納され、デバイス情報１８２０には当該デバイスに関する情報、例えば、割り込み番号やＩ／Ｏアドレス等は格納される。次エントリ１８３０には次のデバイスブロックへのポインタが格納される。

【００３５】ロードオブジェクトリスト１７４０は、ロードオブジェクトブロック１８５０のリストである。ロードオブジェクトブロック１８５０は、カーネルが主記憶にロードしたオブジェクトファイルのデータを保持する。オブジェクトファイル名１８６０は当該ロードオブジェクトブロックで記述されるオブジェクトファイルのファイル名が格納され、オブジェクトアドレス１７８０は当該ロードオブジェクトブロックで記述されるオブジェクトファイルが格納されるカーネル空間のアドレスが格納される。

【００３６】カーネル構成情報アドレス１７３０はカーネル構成情報１９００のアドレスが格納される。

【００３７】図７は、割り込みベクタ管理テーブル２０

００の構造を示す図である。割り込みベクタ管理テーブル２０００は、ＯＳ毎にあり、各割り込み番号に関してその割り込み番号を当該ＯＳが使用するか否かを示す。カーネルは、デバイスドライバの初期化時に割り込み番号の使用を要求した場合にこの割り込みベクタ管理テーブル２０００を調べ、使用中でない場合には要求された割り込み番号を使用中にして、その割り込み番号を使用する権利を与える。

【００３８】図８は、Ｉ／Ｏアドレス管理リスト２１００の構造を示す図である。Ｉ／Ｏアドレス管理リスト２１００は、ＯＳ毎にあり、Ｉ／Ｏアドレスの範囲を示すエントリ２１１０と次エントリへのポインタ２１２０から成り、デバイスドライバの初期化時にＩ／Ｏアドレスの使用を要求した場合にこのＩ／Ｏアドレス管理リスト２１００を調べ、使用中でない場合には当該Ｉ／Ｏアドレスのエントリを追加して、そのＩ／Ｏアドレスを使用する権利を与える。

【００３９】図９は、プロセッサ管理テーブル２１５０の構造を示す図である。プロセッサ管理テーブル２１５０は、ＯＳ毎にあり、各プロセッサに関して当該ＯＳがそのプロセッサを使用するか否かを示す。

【００４０】図１０は、第１のＯＳの初期化処理のフローである。まず、パラメータとして渡されたパラメータテーブル１７００中のロードオブジェクトリスト１７４０を参照して、カーネルローダ１１３１がロードしたオブジェクトファイルの外部参照アドレスを解決する（ステップ２２００）。次に、カーネル構成情報１９００の中の第２のＯＳ１６２０という名前のエントリの第２のＯＳが使用するデータ１６４０から、第２のＯＳのために確保する主記憶の大きさを求め、その分を除いて主記憶リスト１７１０を構成しなおして第２のＯＳの主記憶を確保する（ステップ２２１０）。これにより、第１のＯＳからは第２のＯＳのために確保した領域は見えなくなる。カーネル内部のデータ構造を初期化する（ステップ２２２０）。次に、第１のＯＳが管理する割り込みベクタ管理テーブル２０００およびＩ／Ｏアドレス管理リスト２１００に対して、第２のＯＳが使用するデバイス（カーネル構成情報１９００の中の第２のＯＳ１６２０という名前のエントリの第２のＯＳが使用するデータ１６４０に格納されている）を登録して第２のＯＳのデバイスを予約する（ステップ２２３０）。これにより予約した第２のＯＳのデバイスは第１のＯＳからは利用できなくなる。第１のＯＳが管理するプロセッサ管理テーブル２１５０に対して、第２のＯＳが使用するプロセッサ（カーネル構成情報１９００の中の第２のＯＳ１６２０という名前のエントリの第２のＯＳが使用するデータ１６４０に格納されている）を登録して第２のＯＳのプロセッサを予約する（ステップ２２３５）。これにより予約した第２のＯＳのプロセッサは第１のＯＳからは利用できなくなる。カーネルが直接管理するシステムデバイスを初期化する（ステップ２２４０）。ここでは、第１のＯＳが属する仮想クラスタが専有するプロセッサのうち、この初期化処理を実行中のプロセッサ以外のプロセッサに対して初期化割り込みをかける。更に、割り込み制御装置１０８０の初期化も行う。即ち、カーネル構成情報１９００の中の第２のＯＳ１６２０という名前のエントリの第２のＯＳが使用するデータ１６４０を参照して、第２のＯＳが管理するデバイスの割り込みが、第２のＯＳが専有するプロセッサに送られるように設定する。カーネルローダ１１３１がロードしたオブジェクトファイルを初期化し（ステップ２２５０）、初期プロセスを作成する（ステップ２２６０）。

【００４１】図１１は、第２のＯＳのロード処理のフローである。第２のＯＳの物理メモリ領域は第１のＯＳからはアクセスできないので、第２のＯＳに割り当てた物理メモリ領域を第１のＯＳの仮想空間に一時的にマッピングする（ステップ２３００）。マッピングした領域に第１にＯＳのファイル読み込み手順を利用して第２のＯＳのオブジェクトファイルを読み込む（ステップ２３１０）。次に、第２にＯＳのためのページテーブルを作成する（ステップ２３２０）。この時、第１のＯＳ共有する部分については第２のＯＳからも参照できるようにページテーブルを構築する。第２のＯＳの外部参照を解決する（ステップ２３３０）。そして、ステップ２３００で設定したマッピングを解除する（ステップ２３４０）。ページテーブルアドレスとスタックポインタから成るＯＳコンテキストを設定し（ステップ２３５０）、第２のＯＳが専有するプロセッサに初期化割り込みを送信し（ステップ２３５５）、第２のＯＳの初期化手順を実行する（ステップ２３６０）。最後に、現在の割り込みテーブルを書き換える（ステップ２３７０）。

【００４２】図１２は、割り込み制御装置１０８０を説明する図である。割り込み制御装置１０８０は、各デバイスからの割り込みをどのプロセッサあるいはプロセッサ群へ通知するかを設定する機能を有する。この割り込み制御装置１０８０により、デバイスからの割り込みを、当該デバイスが属する仮想クラスタが専有するプロセッサへ通知することが可能となる。

【００４３】割り込みマスクレジスタ２４４０はデバイスで発生した割り込みをプロセッサに通知してよいかどうかを設定される。これはプロセッサ１０１０からの命令で設定できる。選択装置２４３０は、割り込み信号線１１００から割り込み要求を受けた時と割り込みマスクレジスタ２４４０の内容が変更された時に、選択装置２４３０が記憶している未処理割り込みと割り込みマスクレジスタ２４４０の内容を比較して、割り込みマスクレジスタ２４４０に割り込み可能と設定されていて最も優先度の高い割り込みから順にプロセッサに通知する。

【００４４】割り込み配送テーブル２４１０は、それぞれのデバイスについてどのプロセッサあるいはプロセッ

サ群へ通知するかを示す配送先２４１１と、通知するときの割り込み番号２４１２から成る。この割り込み配送テーブル２４１０の内容はプロセッサの命令により自由に設定できる。

【００４５】割り込み送信装置２４２０は、選択装置２４３０からの信号を受けると、割り込み配送テーブル２４１０の内容にしたがって割り込み通知先と割り込み番号を決定し、その信号を割り込みバス１１０１に送信する。

【００４６】以上により、１台の対称型マルチプロセッサシステム１０００の上に、プロセッサ、主記憶およびデバイスを専有しその上でＯＳが動作する仮想クラスタを複数個構築することが可能となる。

【００４７】以下では、異なる仮想クラスタ上で動作するプロセスが相互に通信する仮想クラスタ間通信について説明する。

【００４８】図１３は、主記憶装置１０５０における仮想クラスタ間通信関連の内容を示す図である。主記憶装置１０５０は、図１に示したように、ＯＳ間共通領域１０５５、第１のＯＳの領域１０５６、および第２のＯＳの領域１０５７から成る。ＯＳ間共通領域１０５５の中には、仮想クラスタ間通信処理部３０１０があり、その中には仮想クラスタ間通信オブジェクト管理表３０００がある。第１のＯＳの領域１０５６および第２のＯＳの領域１０５７には、それぞれ第１のＯＳのプロセス間通信機能３２００および第２のＯＳのプロセス間通信機能３３００がある。これらは、それぞれのＯＳ自身がその上で動作するプロセスに対するプロセス間通信処理を行う部分である。そして、第１のＯＳの領域１０５６および第２のＯＳの領域１０５７には、それぞれ第１のＯＳのプロセス３４００および第２のＯＳのプロセス３５００がある。

【００４９】図１４は、仮想クラスタ間通信オブジェクト管理表３０００の図である。仮想クラスタ間通信オブジェクト管理表３０００は、エントリ有効フラグ３０１０、通信オブジェクトの名前３０２０、第１のＯＳの通信オブジェクトハンドル３０３０、第１のＯＳの通信オブジェクトハンドル有効フラグ３０４０、第２のＯＳの通信オブジェクトハンドル３０５０、第２のＯＳの通信オブジェクトハンドル有効フラグ３０６０から成る。

【００５０】通信オブジェクトの名前３０２０には、仮想クラスタ間通信オブジェクトの名前が格納される。エントリ有効フラグ３０１０は、このエントリが有効か無効かを示すフラグである。

【００５１】第１のＯＳの通信オブジェクトハンドル３０３０には、第１のＯＳのプロセス間通信機能３２００のオブジェクトハンドルを格納し、第１のＯＳの通信オブジェクトハンドル有効フラグ３０４０はそれの有効か無効かを示すフラグである。

【００５２】第２のＯＳの通信オブジェクトハンドル３０５０には、第２のＯＳのプロセス間通信機能３３００のオブジェクトハンドルを格納し、第２のＯＳの通信オブジェクトハンドル有効フラグ３０６０はそれの有効か無効かを示すフラグである。

【００５３】以下では、イベント同期通信（ポスト／ウェイト）の場合を例にとって説明を進めるが、これ以外の通信（メッセージ、セマフォ等）でも同様である。

【００５４】図１５は、第１のＯＳのプロセス３４００が発行したイベント同期通信のオープン要求システムコールの処理のフローである。このシステムコールは、イベントの名前をパラメータに指定してオープン要求を行い、その結果として記述子を得る。以降のイベント同期通信関連のシステムコールでは、この記述子を用いることになる。

【００５５】まず、仮想クラスタ間通信オブジェクト管理表３０００を調べて、エントリ有効フラグ３０１０がＯＮで、かつ、指定されたイベントの名前と通信オブジェクトの名前３０２０が同じであるエントリを探す（ステップ４０１０）。仮想クラスタ間通信オブジェクト管理表３０００にこのようなエントリがあれば、既にこの名前のイベントはオープンされていることになる。そこで、このエントリについて、第１のＯＳの通信オブジェクトハンドル有効フラグ３０４０を調べ（ステップ４０９０）、ＯＮなら、そのエントリの記述子を返して終了する（ステップ４１００）。ＯＦＦなら、第１のＯＳのプロセス間通信機能３２００のオープンのシステムコールをこのイベントの名前で発行することにより、第１のＯＳのプロセス間通信機能３２００におけるオブジェクトハンドルを獲得し（ステップ４１１０）、獲得したオブジェクトハンドルを第１のＯＳの通信オブジェクトハンドル３０３０にセットし（ステップ４１２０）、第１のＯＳの通信オブジェクトハンドル有効フラグ３０４０をＯＮにし（ステップ４１３０）、そのエントリの記述子を返して終了する（ステップ４１４０）。

【００５６】一方、ステップ４０１０で、エントリ有効フラグ３０１０がＯＮで、かつ、指定されたイベントの名前と通信オブジェクトの名前３０２０が同じであるエントリが、仮想クラスタ間通信オブジェクト管理表３０００になければ、エントリ有効フラグ３０１０がＯＦＦのエントリを探す（ステップ４０２０）。なければ、仮想クラスタ間通信オブジェクト管理表３０００のすべてのエントリが使用中であることになるのでエラーリターンする（ステップ４０３０）。エントリ有効フラグ３０１０がＯＦＦのエントリがあれば、第１のＯＳのプロセス間通信機能３２００のオープンのシステムコールをこのイベントの名前で発行することにより、第１のＯＳのプロセス間通信機能３２００のオブジェクトハンドルを獲得し（ステップ４０４０）、獲得したオブジェクトハンドルを第１のＯＳの通信オブジェクトハンドル３０３０にセットし（ステップ４０５０）、第１のＯＳの通信

オブジェクトハンドル有効フラグ３０４０をＯＮにし（ステップ４０６０）、パラメータで指定されたイベントの名前を通信オブジェクトの名前３０２０に格納し（ステップ４０６５）、エントリ有効フラグ３０１０をＯＮにして（ステップ４０７０）、そのエントリの記述子を返して終了する（ステップ４０８０）。

【００５７】図１６は、第１のＯＳのプロセス３４００が発行したイベント同期通信のウェイト要求システムコールの処理のフローである。このシステムコールは、記述子をパラメータに指定してウェイト要求を行う。まず、当該ウェイト要求において指定された記述子に対応するエントリを仮想クラスタ間通信オブジェクト管理表３０００からみつけ、そのエントリにおける第１のＯＳの通信オブジェクトハンドル３０３０を用いて、第１のＯＳのプロセス間通信機能３２００のウェイト要求のシステムコールを発行し（ステップ４２１０）、リターンする（ステップ４２２０）。

【００５８】図１７は、第１のＯＳのプロセス３４００が発行したイベント同期通信のポスト要求システムコールの処理のフローである。このシステムコールは、記述子をパラメータに指定してポスト要求を行う。まず、第２のＯＳへコンテキスト切替えを行い（ステップ４３１０）、当該ポスト要求において指定されたイベント記述子に対応するエントリを仮想クラスタ間通信オブジェクト管理表３０００からみつけ、そのエントリにおける第２のＯＳの通信オブジェクトハンドル３０５０を用いて、第２のＯＳのプロセス間通信機能３３００のポスト要求のシステムコールを発行する（ステップ４３２０）。そして、第１のＯＳへコンテキスト切替えを行い（ステップ４３３０）、リターンする（ステップ４３４０）。

【００５９】図１８は、第１のＯＳのプロセス３４００が発行したイベント同期通信のクローズ要求システムコールの処理のフローである。このシステムコールは、記述子をパラメータに指定してクローズ要求を行う。まず、当該クローズ要求において指定された記述子に対応するエントリを仮想クラスタ間通信オブジェクト管理表３０００からみつけ、そのエントリにおける第１のＯＳの通信オブジェクトハンドル３０３０を用いて、第１のＯＳのプロセス間通信機能３２００のクローズ要求のシステムコールを発行し（ステップ４４１０）、第１のＯＳの通信オブジェクトハンドル有効フラグ３０４０をＯＦＦにする（ステップ４４２０）。第２のＯＳの通信オブジェクトハンドル有効フラグ３０６０がＯＦＦなら、エントリ有効フラグ３０１０をＯＦＦにして当該エントリを無効にして（ステップ４４４０）、リターンする（ステップ４４５０）。一方、第２のＯＳの通信オブジェクトハンドル有効フラグ３０６０がＯＮなら、リターンする（ステップ４４５０）。

【００６０】図１９は、第２のＯＳのプロセス３５００が発行したイベント同期通信のオープン要求システムコールの処理のフローである。このシステムコールは、イベントの名前をパラメータに指定してオープン要求を行い、その結果として記述子を得る。以降のイベント同期通信関連のシステムコールでは、この記述子を用いることになる。

【００６１】まず、仮想クラスタ間通信オブジェクト管理表３０００を調べて、エントリ有効フラグ３０１０がＯＮで、かつ、指定されたイベントの名前と通信オブジェクトの名前３０２０が同じであるエントリを探す（ステップ４５１０）。仮想クラスタ間通信オブジェクト管理表３０００にこのようなエントリがあれば、既にこの名前のイベントはオープンされていることになる。そこで、このエントリについて、第２のＯＳの通信オブジェクトハンドル有効フラグ３０６０を調べ（ステップ４５９０）、ＯＮなら、そのエントリの記述子を返して終了する（ステップ４６００）。ＯＦＦなら、第２のＯＳのプロセス間通信機能３３００のオープンのシステムコールをこのイベントの名前で発行することにより、第２のＯＳのプロセス間通信機能３３００のオブジェクトハンドルを獲得し（ステップ４６１０）、獲得したオブジェクトハンドルを第２のＯＳの通信オブジェクトハンドル３０５０にセットし（ステップ４６２０）、第２のＯＳの通信オブジェクトハンドル有効フラグ３０６０をＯＮにし（ステップ４６３０）、そのエントリの記述子を返して終了する（ステップ４６４０）。

【００６２】一方、ステップ４０１０で、エントリ有効フラグ３０１０がＯＮで、かつ、指定されたイベントの名前と通信オブジェクトの名前３０２０が同じであるエントリが、仮想クラスタ間通信オブジェクト管理表３０００になければ、エントリ有効フラグ３０１０がＯＦＦのエントリを探す（ステップ４５２０）。なければ、仮想クラスタ間通信オブジェクト管理表３０００のすべてのエントリが使用中であることになるのでエラーリターンする（ステップ４５３０）。エントリ有効フラグ３０１０がＯＦＦのエントリがあれば、第２のＯＳのプロセス間通信機能３３００のオープンのシステムコールをこのイベントの名前で発行することにより、第２のＯＳのプロセス間通信機能３３００のオブジェクトハンドルを獲得し（ステップ４５４０）、獲得したオブジェクトハンドルを第２のＯＳの通信オブジェクトハンドル３０５０にセットし（ステップ４５５０）、第２のＯＳの通信オブジェクトハンドル有効フラグ３０６０をＯＮにし（ステップ４５６０）、パラメータで指定されたイベントの名前を通信オブジェクトの名前３０２０に格納し（ステップ４５６５）、エントリ有効フラグ３０１０をＯＮにして（ステップ４５７０）、そのエントリの記述子を返して終了する（ステップ４５８０）。

【００６３】図２０は、第２のＯＳのプロセス３５００が発行したイベント同期通信のウェイト要求システムコ

ールの処理のフローである。このシステムコールは、記述子をパラメータに指定してウェイト要求を行う。まず、当該ウェイト要求において指定された記述子に対応するエントリを仮想クラスタ間通信オブジェクト管理表３０００からみつけ、そのエントリにおける第２のＯＳの通信オブジェクトハンドル３０５０を用いて、第２のＯＳのプロセス間通信機能３３００のウェイト要求のシステムコールを発行し（ステップ４７１０）、リターンする（ステップ４７２０）。

【００６４】図２１は、第２のＯＳのプロセス３５００が発行したイベント同期通信のポスト要求システムコールの処理のフローである。このシステムコールは、記述子をパラメータに指定してポスト要求を行う。まず、第１のＯＳへコンテキスト切替えを行い（ステップ４８１０）、当該ポスト要求において指定された記述子に対応するエントリを仮想クラスタ間通信オブジェクト管理表３０００からみつけ、そのエントリにおける第１のＯＳの通信オブジェクトハンドル３０３０を用いて、第１のＯＳのプロセス間通信機能３２００のポスト要求のシステムコールを発行する（ステップ４８２０）。そして、第２のＯＳへコンテキスト切替えを行い（ステップ４８３０）、リターンする（ステップ４８４０）。

【００６５】図２２は、第２のＯＳのプロセス３５００が発行したイベント同期通信のクローズ要求システムコールの処理のフローである。このシステムコールは、記述子をパラメータに指定してクローズ要求を行う。まず、当該クローズ要求において指定された記述子に対応するエントリを仮想クラスタ間通信オブジェクト管理表３０００からみつけ、そのエントリにおける第２のＯＳの通信オブジェクトハンドル３０５０を用いて、第２のＯＳのプロセス間通信機能３３００のクローズ要求のシステムコールを発行し（ステップ４９１０）、第２のＯＳの通信オブジェクトハンドル有効フラグ３０６０をＯＦＦにする（ステップ４９２０）。第１のＯＳの通信オブジェクトハンドル有効フラグ３０４０がＯＦＦなら、エントリ有効フラグ３０１０をＯＦＦにして当該エントリを無効にして（ステップ４９４０）、リターンする（ステップ４９５０）。一方、第１のＯＳの通信オブジェクトハンドル有効フラグ３０４０がＯＮなら、リターンする（ステップ４９５０）。

【００６６】以上の図１５から図２２に示した処理により、ウェイトしている第１のＯＳのプロセス３４００を第２のＯＳのプロセス３５００がポストし、ウェイトしている第２のＯＳのプロセス３５００を第１のＯＳのプロセス３４００がポストすることが可能となる。

【００６７】

【発明の効果】本発明では、第１の仮想クラスタのオペレーティングシステム（第１のオペレーティングシステム）の起動時に、他の仮想クラスタが利用するハードウェア資源を他の仮想クラスタのために予約して第１のオペレーティングシステムの管理対象から外して第１のオペレーティングシステムからはアクセス不可能にし、これらのオペレーティングシステムから独立したプログラムが外部からの割り込みを扱い、どのオペレーティングシステムで処理するかを決定し、割り込みハンドラを起動することにより、１台の対称型マルチプロセッサシステム（ＳＭＰ）において複数個の仮想クラスタを特別なハードウェアなしに構成することが可能となる。

【００６８】これにより、高多重の対称型マルチプロセッサシステム（ＳＭＰ）において、排他制御オーバヘッドを小さくする効果がある。また、高多重の対称型マルチプロセッサシステムにおいて、キャッシュミス、およびコヒーレントトランザクションを減らす効果もある。

【００６９】また、本発明では、仮想クラスタ間通信オブジェクトの名前と、第１のオペレーティングシステムのプロセス間通信機能におけるプロセス間通信オブジェクトハンドルと、第２のオペレーティングシステムのプロセス間通信機能におけるプロセス間通信オブジェクトハンドルとの対応管理手段を設ける。この対応管理手段により、第１のオペレーティングシステム上のプロセスおよび第２のオペレーティングシステム上のプロセスから発行された仮想クラスタ間通信オブジェクトの名前を指定した通信要求を、第１のオペレーティングシステムあるいは第２のオペレーティングシステムのプロセス間通信機能への要求へと変換できる。これにより、例えば、イベント同期によるプロセス間通信の場合では、ウェイトしている第１のオペレーティングシステム上のプロセスを第２のオペレーティングシステム上のプロセスがポストし、ウェイトしている第２のオペレーティングシステム上のプロセスを第１のオペレーティングシステム上のプロセスがポストすることが可能となる。

【００７０】これにより、仮想クラスタ上のプロセス同士が連携して処理を進行することが可能となる効果がある。

【図面の簡単な説明】

【図１】本発明の実施例を実施する計算機システムである対称型マルチプロセッサシステム１０００の構成を示す図である。

【図２】ページテーブル１０５２の構成を示す図である。

【図３】割り込みテーブル１０５１の構成を示す図である。

【図４】計算機のブート時に実行されるカーネルローダ１１３１の処理のフローである。

【図５】カーネル構成情報ファイル１６００の内容を示す図である。

【図６】主記憶リスト１７１０、デバイスリスト１７２０、ロードオブジェクトリスト１７４０およびパラメータテーブル１７００の構成を示す図である。

【図７】割り込みベクタ管理テーブル２０００の構造を

示す図である。
【図８】Ｉ／Ｏアドレス管理リスト２１００の構造を示す図である。
【図９】プロセッサ管理テーブル２１５０の構造を示す図である。
【図１０】第１のＯＳの初期化処理のフローである。
【図１１】第２のＯＳのロード処理のフローである。
【図１２】割り込み制御装置１０８０を説明する図である。
【図１３】主記憶装置１０５０における仮想クラスタ間通信関連の内容を示す図である。
【図１４】仮想クラスタ間通信オブジェクト管理表３０００の図である。
【図１５】第１のＯＳのプロセス３４００が発行したイベント同期通信のオープン要求システムコールの処理のフローである。
【図１６】第１のＯＳのプロセス３４００が発行したイベント同期通信のウェイト要求システムコールの処理のフローである。
【図１７】第１のＯＳのプロセス３４００が発行したイベント同期通信のポスト要求システムコールの処理のフローである。
【図１８】第１のＯＳのプロセス３４００が発行したイベント同期通信のクローズ要求システムコールの処理のフローである。
【図１９】第２のＯＳのプロセス３５００が発行したイベント同期通信のオープン要求システムコールの処理のフローである。
【図２０】第２のＯＳのプロセス３５００が発行したイベント同期通信のウェイト要求システムコールの処理のフローである。
【図２１】第２のＯＳのプロセス３５００が発行したイベント同期通信のポスト要求システムコールの処理のフローである。
【図２２】第２のＯＳのプロセス３５００が発行したイベント同期通信のクローズ要求システムコールの処理のフローである。
【符号の説明】
１０００：対称型マルチプロセッサシステム（ＳＭＰ）、１０１０：プロセッサ、１０１１：プロセッサ、１０２０：演算装置、１０３０：割り込みテーブルレジスタ、１０４０：ページテーブルレジスタ、１０４５：アドレス変換装置、１０５０：主記憶装置、１０５１：割り込みテーブル、１０５２：ページテーブル、１０５７：第２のＯＳの領域、１０６０：クロック、１０７０：ブートＲＯＭ、１０８０：割り込み制御装置、１０９０：バス、１１００：割り込み信号線、１１０１：割り込みバス、１１１０：キーボード、１１２０：モニタ装置、１１３０：磁気ディスク装置、１１３１：カーネルローダ、１１４０：その他の外部装置、１２００：仮想ページ番号、１２１０：有効ビット、１２２０：物理ページ番号、１３００：割り込み番号、１３１０：割り込みハンドラアドレス、１６００：カーネル構成情報ファイル１７００：パラメータテーブル、１７１０：主記憶リスト、１７２０：デバイスリスト、１７３０：カーネル構成情報アドレス、１７４０：ロードオブジェクトリスト、１７５０：主記憶ブロック、１８００：デバイスブロック、１８５０：ロードオブジェクトブロック、１９００：カーネル構成情報、２０００：割り込みベクタ管理テーブル、２１００：Ｉ／Ｏアドレス管理リスト、２１５０：プロセッサ管理テーブル、２４１０：割り込み配送テーブル、２４２０：割り込み送信装置、２４３０：選択装置、２４４０：割り込みマスクレジスタ、３０００：仮想クラスタ間通信オブジェクト管理表、３０１０：仮想クラスタ間通信処理部、３２００：第１のＯＳのプロセス間通信機能、３３００：第２のＯＳのプロセス間通信機能、３４００：第１のＯＳのプロセス、３５００：第２のＯＳのプロセス。

【図２】

図2

ページテーブル

1200 仮想ページ番号	1210 有効ビット	1220 物理ページ番号
0	1	0
1	1	3
2	0	28
3	1	15
4	0	41
⋮	⋮	⋮

1052

【図３】

図3

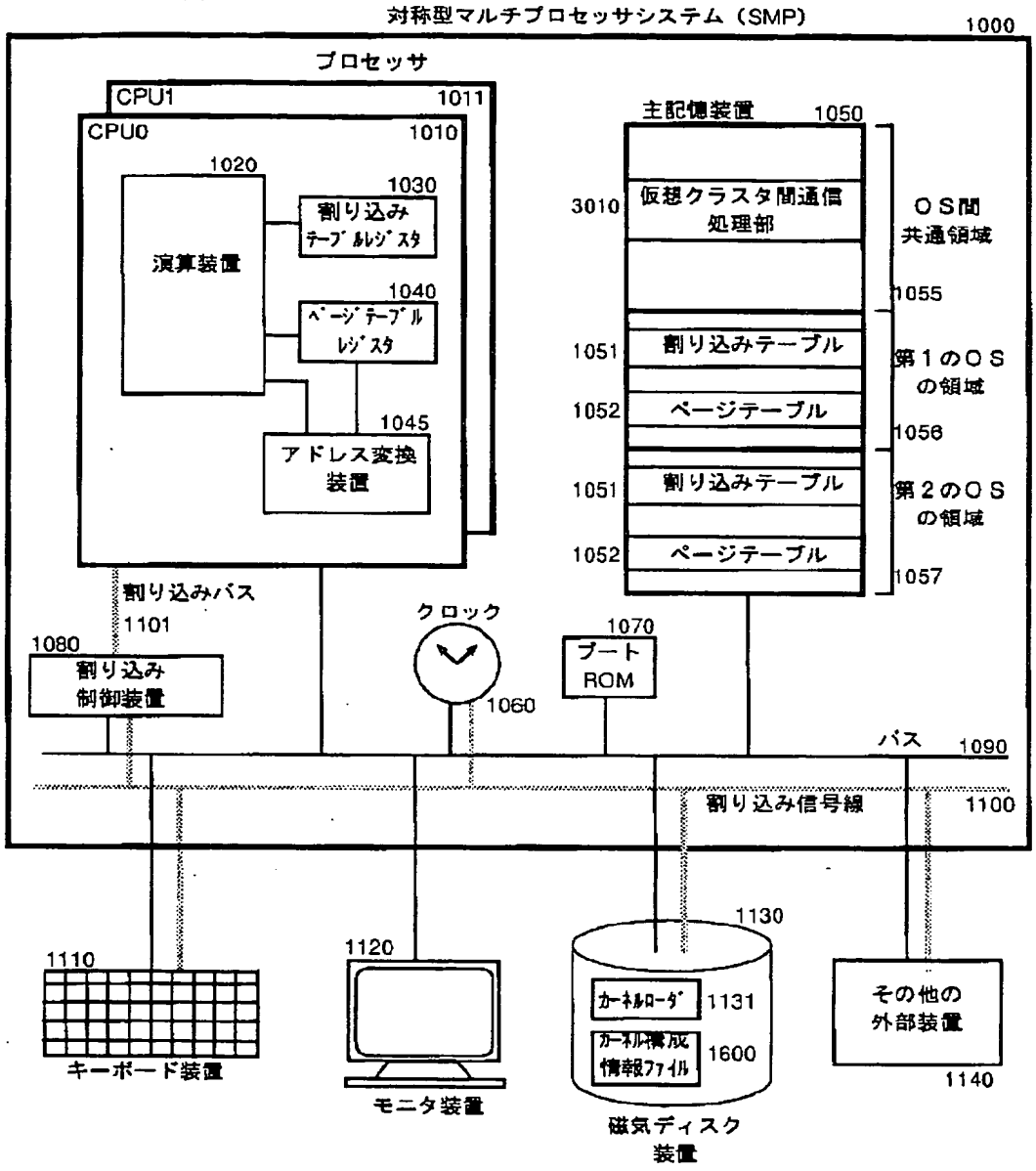
割り込みテーブル

1300 割り込み番号	1310 割り込みハンドラアドレス
0	200
1	220
2	240
3	260
4	280
⋮	⋮

1051

【図１】

図１



【図５】

図５

カーネル構成情報ファイル　1600

名前	内容
1610 オブジェクトファイル	kernel, driver 1630
1620 第２のOS	第２のOSが使用するデータ 1640

【図９】

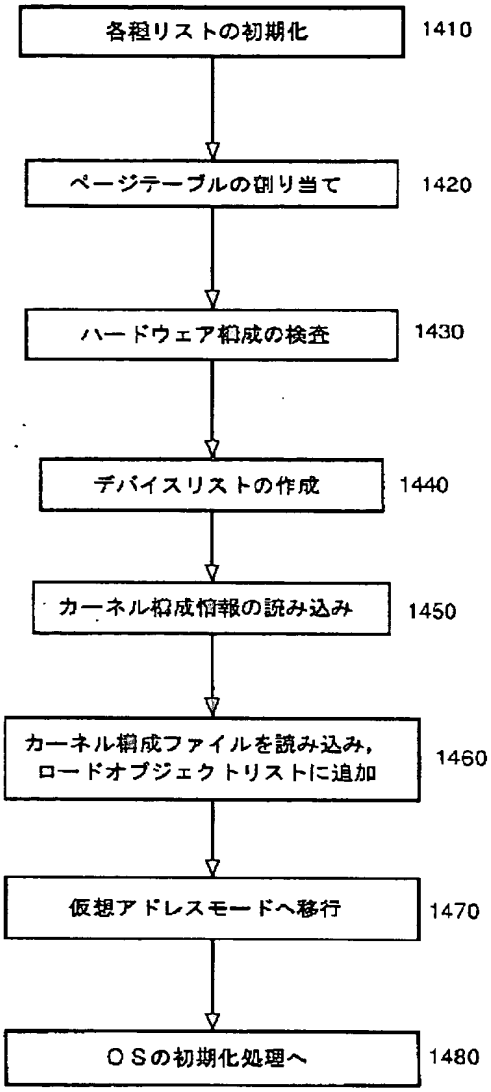
図９

プロセッサ管理テーブル　2150

プロセッサ	使用中
CPU0	
CPU1	

【図４】

図４



【図７】

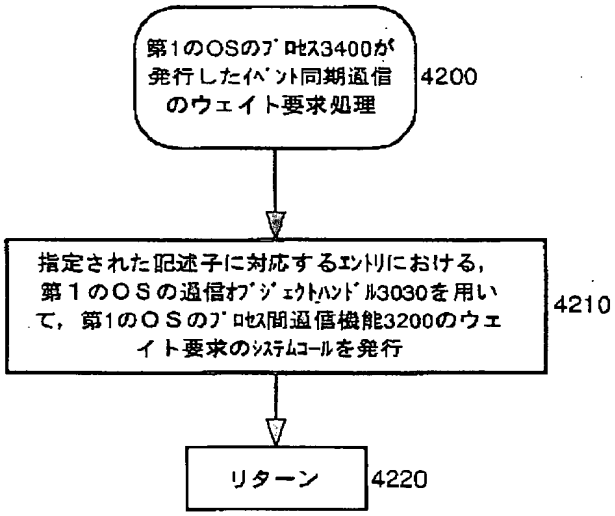
図７

割り込みベクタ管理テーブル　2000

割り込み番号	使用中
0	
1	
2	
3	
4	
・・・	

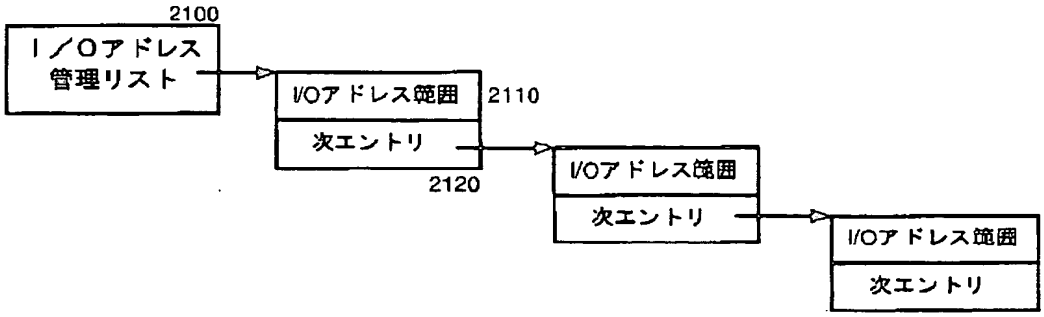
【図１６】

図１６



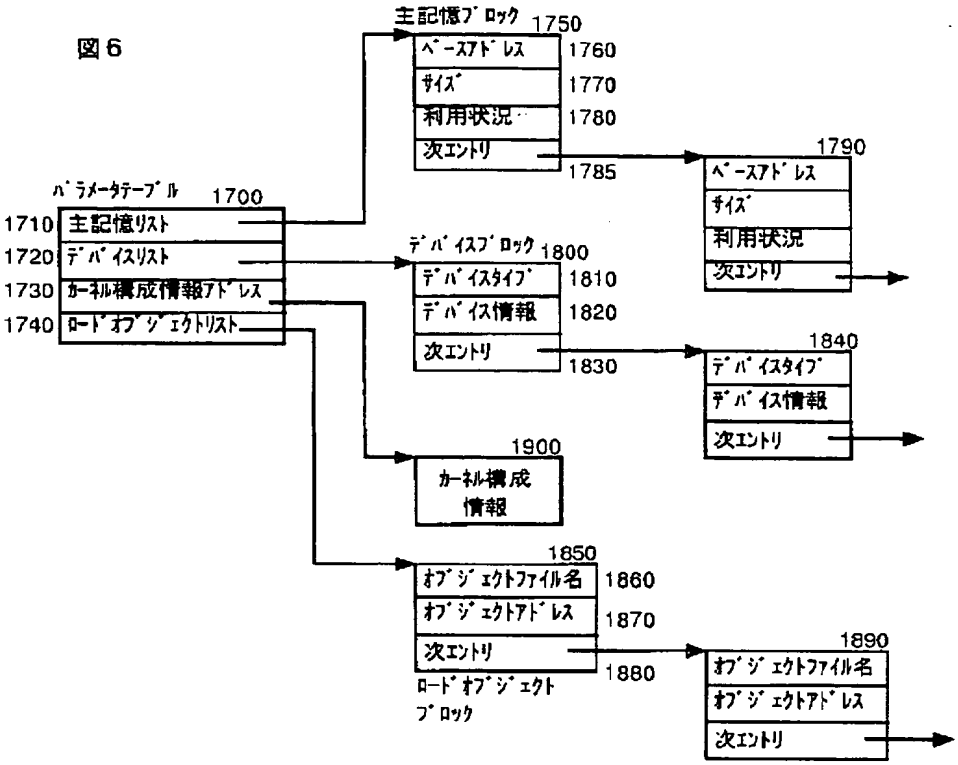
【図８】

図８



【図６】

図６



【図１４】

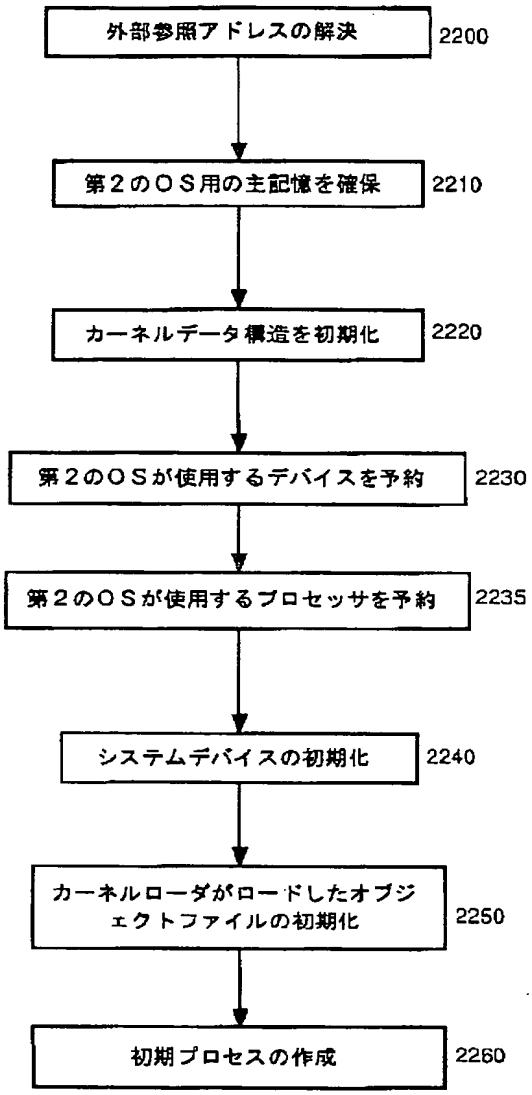
図１４

仮想クラスタ間通信オブジェクト管理表　　3000

エントリ有効フラグ	通信オブジェクトの名前	第１のOSの通信オブジェクトハンドル	第１のOSの通信オブジェクトハンドル有効フラグ	第２のOSの通信オブジェクトハンドル	第２のOSの通信オブジェクトハンドル有効フラグ
3010	3020	3030	3040	3050	3060

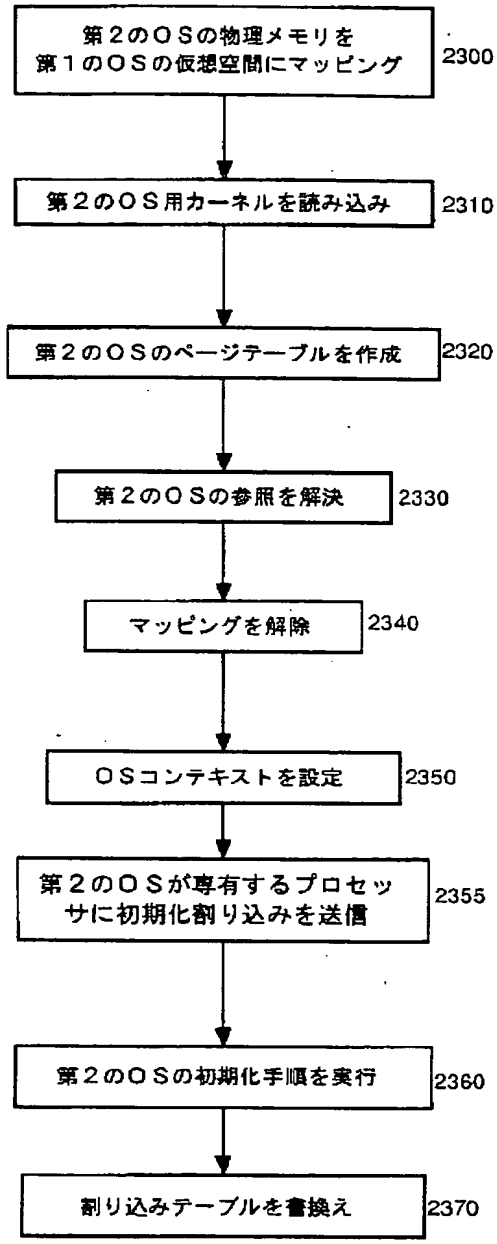
【図１０】

図１０



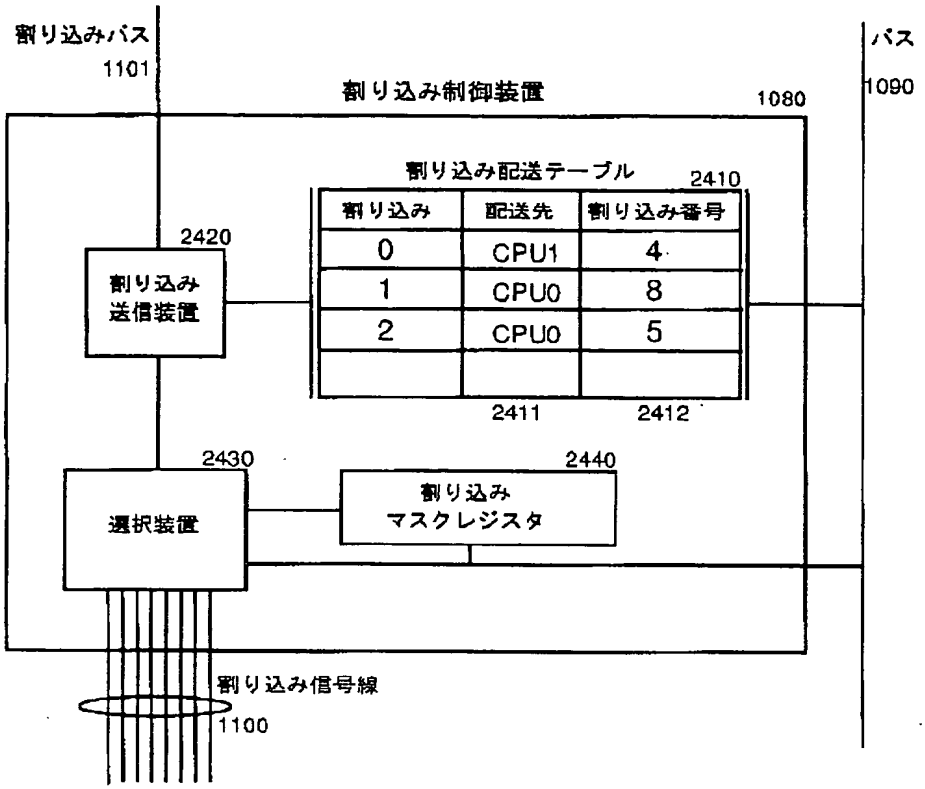
【図１１】

図１１



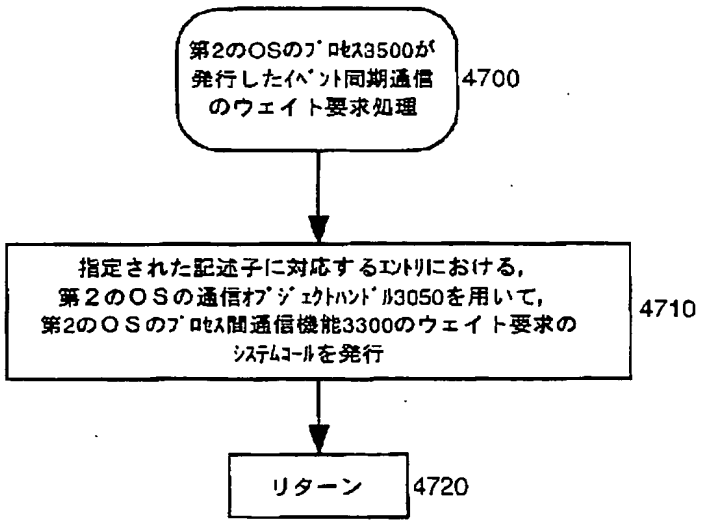
【図１２】

図１２



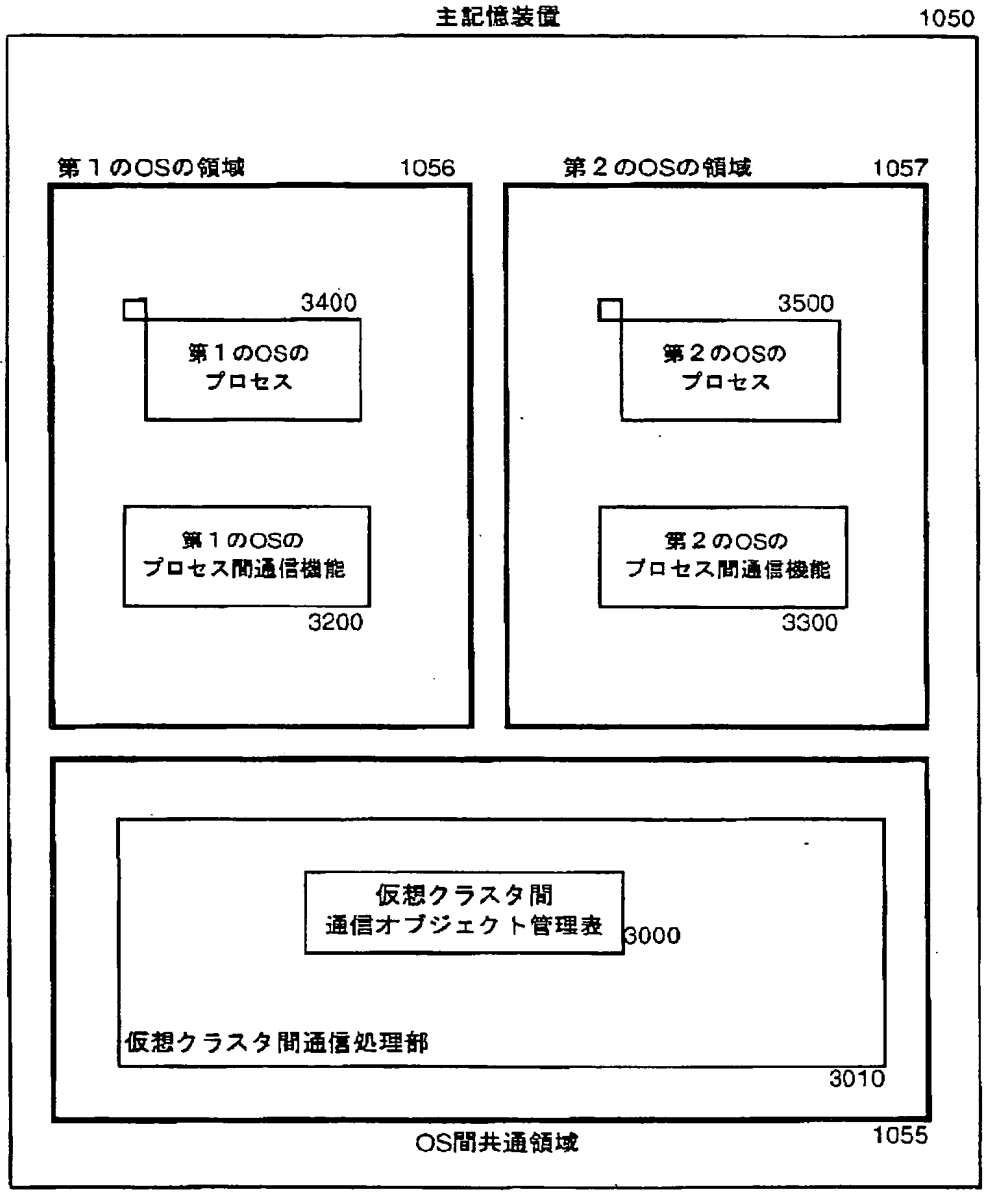
【図２０】

図２０



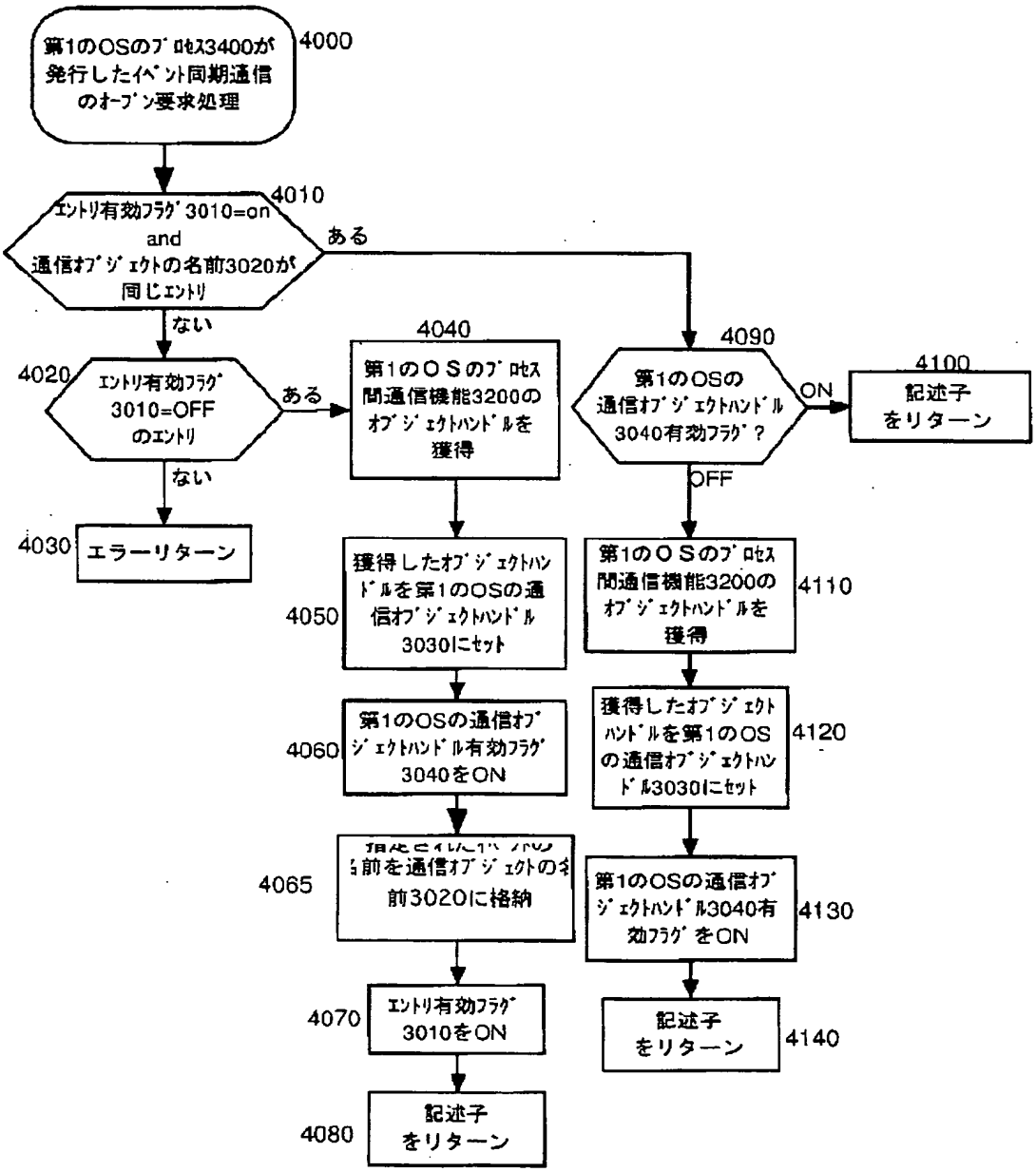
【図１３】

図１３


主記憶装置
1050
第１のOSの領域
1056
第２のOSの領域
1057
3400
第１のOSの
プロセス
3500
第２のOSの
プロセス
第１のOSの
プロセス間通信機能
3200
第２のOSの
プロセス間通信機能
3300
仮想クラスタ間
通信オブジェクト管理表
3000
仮想クラスタ間通信処理部
3010
OS間共通領域
1055

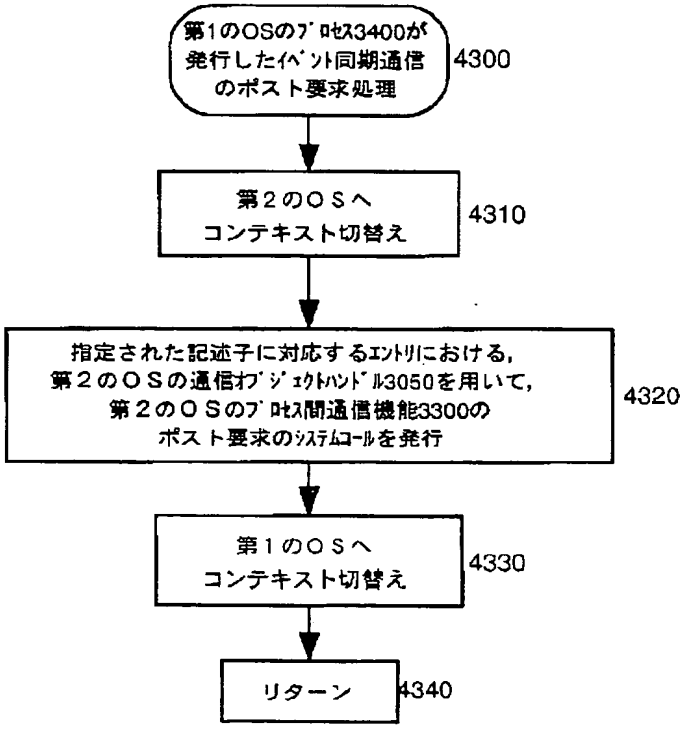
【図１５】

図１５


第1のOSのﾌﾟﾛｾｽ3400が発行したｲﾍﾞﾝﾄ同期通信のｵｰﾌﾟﾝ要求処理
4000
ｴﾝﾄﾘ有効ﾌﾗｸﾞ3010=on and 通信ｵﾌﾞｼﾞｪｸﾄの名前3020が同じｴﾝﾄﾘ
4010
ある
ない
ｴﾝﾄﾘ有効ﾌﾗｸﾞ3010=OFFのｴﾝﾄﾘ
4020
ある
ない
4030
エラーリターン
4040
第1のＯＳのﾌﾟﾛｾｽ間通信機能3200のｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを獲得
4050
獲得したｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを第1のOSの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3030にｾｯﾄ
4060
第1のOSの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ有効ﾌﾗｸﾞ3040をON
4065
前3020に格納
4070
ｴﾝﾄﾘ有効ﾌﾗｸﾞ3010をON
4080
記述子をリターン
4090
第1のOSの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3040有効ﾌﾗｸﾞ？
ON
4100
記述子をリターン
OFF
4110
第1のＯＳのﾌﾟﾛｾｽ間通信機能3200のｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを獲得
4120
獲得したｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを第1のOSの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3030にｾｯﾄ
4130
第1のOSの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3040有効ﾌﾗｸﾞをON
4140
記述子をリターン

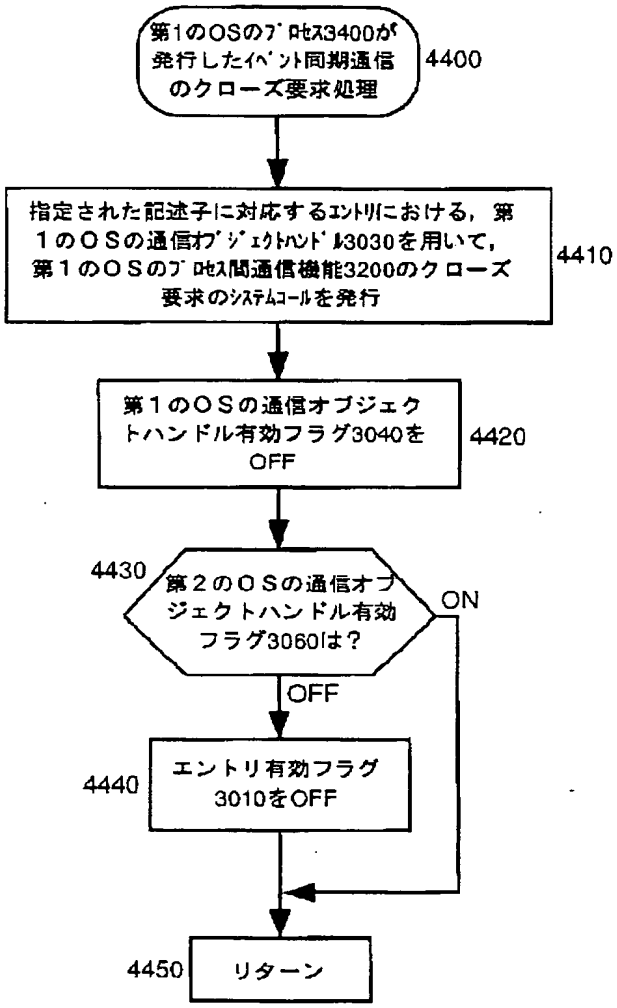
【図１７】

図１７


第1のOSのﾌﾟﾛｾｽ3400が
発行したｲﾍﾞﾝﾄ同期通信
のポスト要求処理
4300
第２のＯＳへ
コンテキスト切替え
4310
指定された記述子に対応するｴﾝﾄﾘにおける，
第２のＯＳの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3050を用いて，
第２のＯＳのﾌﾟﾛｾｽ間通信機能3300の
ポスト要求のｼｽﾃﾑｺｰﾙを発行
4320
第１のＯＳへ
コンテキスト切替え
4330
リターン
4340

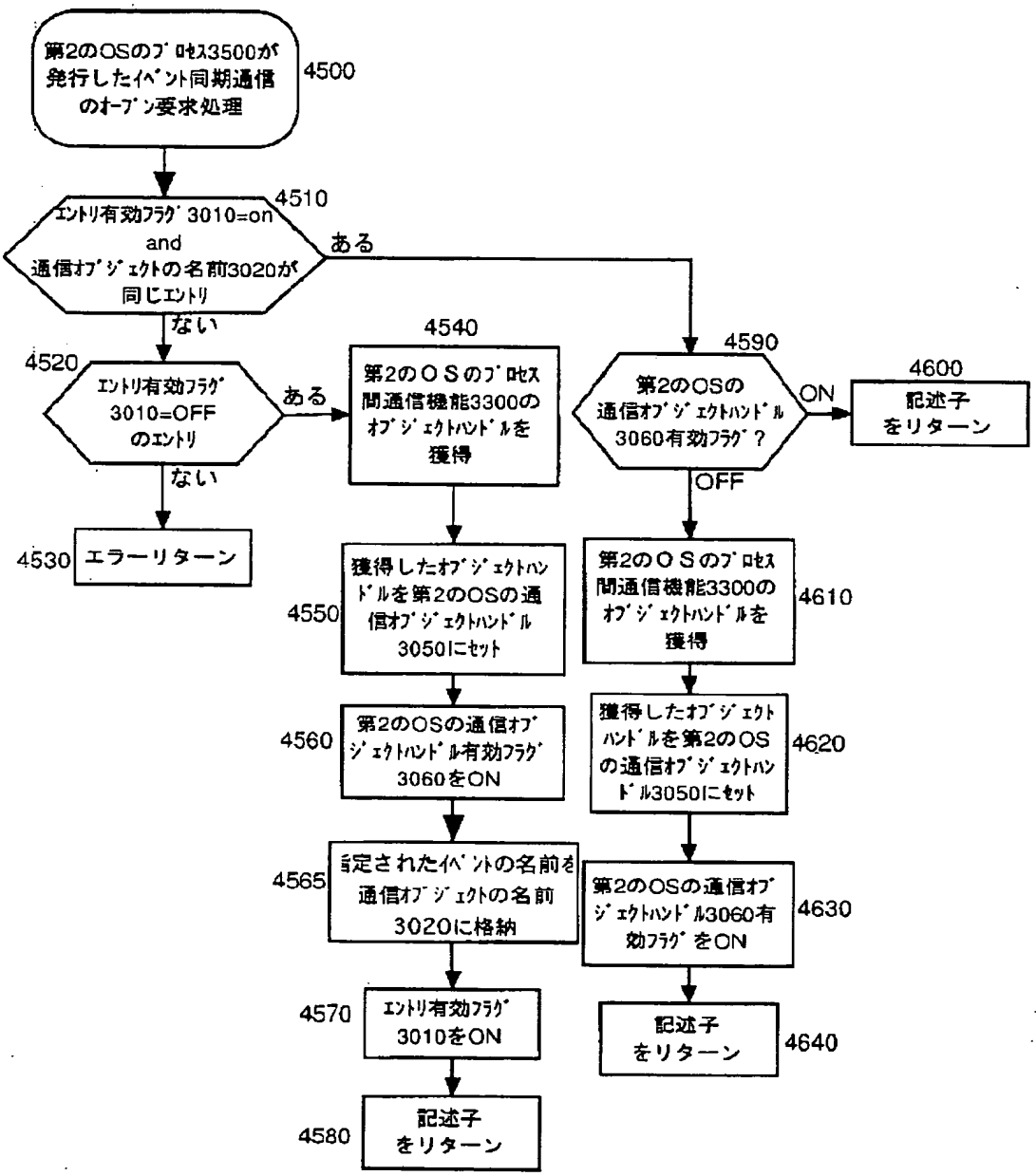
【図１８】

図１８


第1のOSのﾌﾟﾛｾｽ3400が発行したｲﾍﾞﾝﾄ同期通信のクローズ要求処理
4400
指定された記述子に対応するｴﾝﾄﾘにおける，第１のＯＳの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3030を用いて，第１のＯＳのﾌﾟﾛｾｽ間通信機能3200のクローズ要求のｼｽﾃﾑｺｰﾙを発行
4410
第１のＯＳの通信オブジェクトハンドル有効フラグ3040をOFF
4420
4430
第２のＯＳの通信オブジェクトハンドル有効フラグ3060は？
ON
OFF
4440
エントリ有効フラグ3010をOFF
4450
リターン

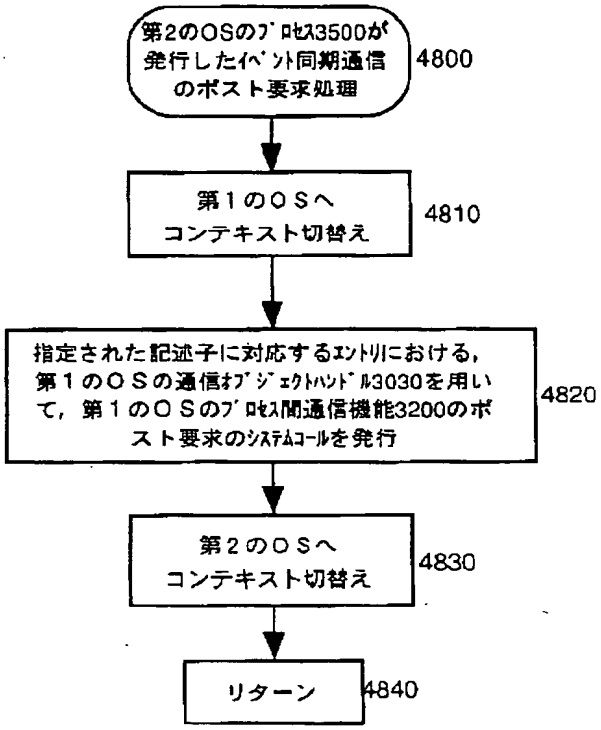
【図１９】

図１９


第2のOSのﾌﾟﾛｾｽ3500が発行したｲﾍﾞﾝﾄ同期通信のｵｰﾌﾟﾝ要求処理
4500
ｴﾝﾄﾘ有効ﾌﾗｸﾞ3010=on and 通信ｵﾌﾞｼﾞｪｸﾄの名前3020が同じｴﾝﾄﾘ
4510
ある
ない
4520
ｴﾝﾄﾘ有効ﾌﾗｸﾞ3010=OFFのｴﾝﾄﾘ
ある
ない
4530
エラーリターン
4540
第2のＯＳのﾌﾟﾛｾｽ間通信機能3300のｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを獲得
4550
獲得したｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを第2のOSの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3050にｾｯﾄ
4560
第2のOSの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ有効ﾌﾗｸﾞ3060をON
4565
設定されたｲﾍﾞﾝﾄの名前を通信ｵﾌﾞｼﾞｪｸﾄの名前3020に格納
4570
ｴﾝﾄﾘ有効ﾌﾗｸﾞ3010をON
4580
記述子をリターン
4590
第2のOSの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3060有効ﾌﾗｸﾞ？
ON
4600
記述子をリターン
OFF
4610
第2のＯＳのﾌﾟﾛｾｽ間通信機能3300のｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを獲得
4620
獲得したｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを第2のOSの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3050にｾｯﾄ
4630
第2のOSの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3060有効ﾌﾗｸﾞをON
4640
記述子をリターン

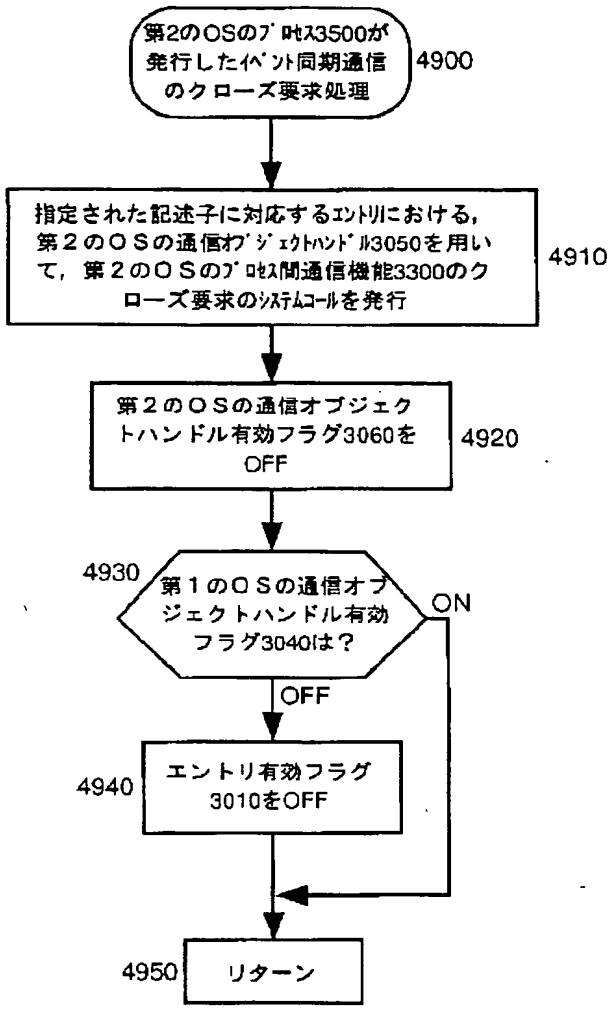
【図２１】

図２１


第2のOSのﾌﾟﾛｾｽ3500が発行したｲﾍﾞﾝﾄ同期通信のポスト要求処理
4800
第１のＯＳへコンテキスト切替え
4810
指定された記述子に対応するｴﾝﾄﾘにおける，第１のＯＳの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3030を用いて，第１のＯＳのﾌﾟﾛｾｽ間通信機能3200のポスト要求のｼｽﾃﾑｺｰﾙを発行
4820
第２のＯＳへコンテキスト切替え
4830
リターン
4840

【図２２】

図２２


第2のOSのﾌﾟﾛｾｽ3500が発行したｲﾍﾞﾝﾄ同期通信のクローズ要求処理
4900
指定された記述子に対応するｴﾝﾄﾘにおける，第２のＯＳの通信ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3050を用いて，第２のＯＳのﾌﾟﾛｾｽ間通信機能3300のクローズ要求のｼｽﾃﾑｺｰﾙを発行
4910
第２のＯＳの通信オブジェクトハンドル有効フラグ3060をOFF
4920
4930
第１のＯＳの通信オブジェクトハンドル有効フラグ3040は？
ON
OFF
4940
エントリ有効フラグ3010をOFF
4950
リターン